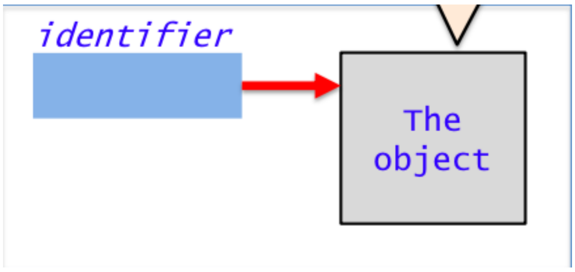
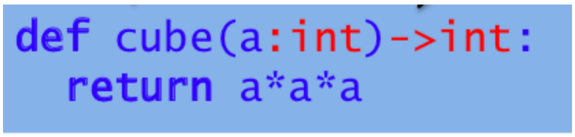
[Lecture 4] More on functions Modules



Binding between **identifier** (reference, name) and the **referenced object**(class, function, variable...) **is weak:** it can be changed dynamically, while program runs

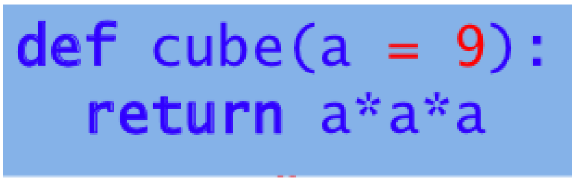
del fun - removes reference from scope



```
def cube(a:int)->int:
  return a*a*a
```

Type annotations purely for documenting functions. Python don't check type.

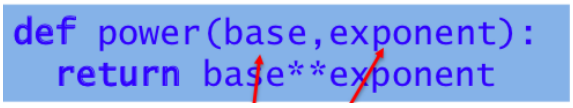
help(cube) → cube(a: int) → int



```
def cube(a = 9):
  return a*a*a
```

Параметры с defaulf agruments указываются в конце списка параметров функции.

Функцию можно вызвать только указав точное число требуемых параметров. Но возможно задать **default arguments**, которые будут использоваться, если ничего не задать



```
def power(base,exponent):
  return base**exponent
```

Positional arguments - при вызове все аргументы указываются в порядке объявления. (power('base', 'exponent'))

Named arguments - при вызове можно указать аргументы с их именами. (power(exponent=10,base=7))

```
def process(action, *items):
```

Ex: print()

Arbitrary number of arguments (≥0)(technically, this is a **tuple**)

How to call:

- process(quad,1,2,3,4,5,6)
- **values = [1,2,3,4,5,6]** ⇒ process(quad,*values)
- **intro(Firstname="Sita", Lastname="Sharma")** ⇒ def intro(**data):

Module - collection of functions and other entities (stored as file on disk, module gets the name of file main.py)

Library - collection of modules (math, os, sys, random… → standard Python library)

Use:

- ▼ import module

  its **name** – but not the names of its entities – is known within the program that uses the module.

- ▼ from module import fun1, fun2

  The module is not imported, only **entities**

- ▼ from module import *

  All entities imported except module itself and names started with underscore '_'

- ▼ import FibCollection as FC

  просто упрощение, также можно сделать from math import sin as s

Modules are objects!

References actions:

- x = myLib → help(x) = help(myLib)
- pass to function as argument
- return as function result

Modules and file paths

- Precompiledmodules;.pyand.pycfiles
- Standardmodules:sys,mathetc.
- Submodules and packages